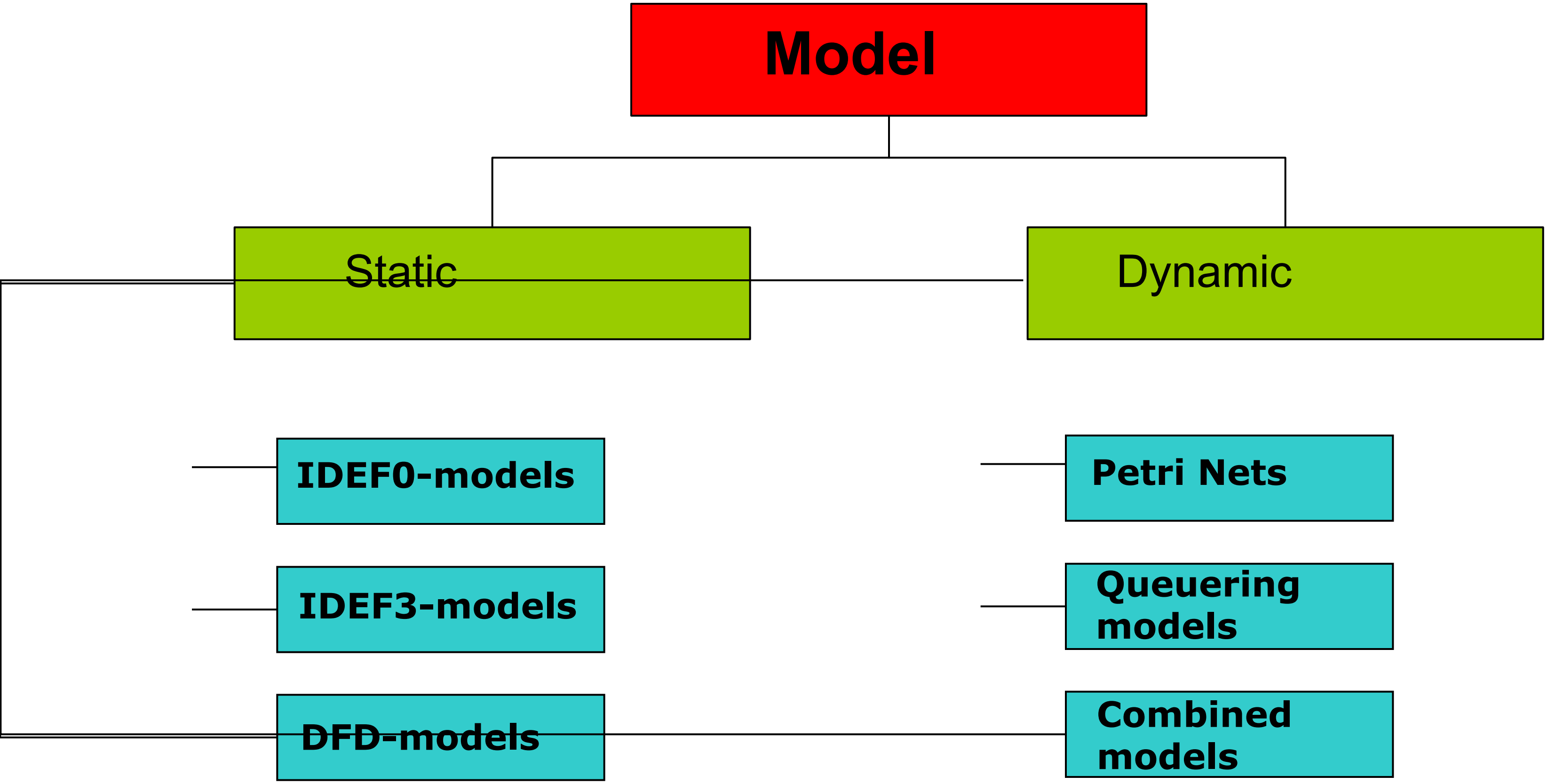
SIMULATION

Part 1

Model classification



Petri Nets

□ Petri nets were developed in the early 1960s by Carl Adam Petri in his Ph.D. dissertation "*Kommunikation mit Automaten*" *(Automata Connection),* Institut für instrumentelle Mathematik, Bonn, 1962

□ They are useful for modelling concurrent, distributed, asynchronous behaviour in a system

What is a Petri net?

1. A bipartite graph **G(V,E)** where

- V = P U T

  P is the set of places (shown as circles)

  T is the set of transitions (shown as vertical bars);

- E is the set of edges between P and T

2. Marking function M. Given μ *belongs to* M, each μ is a function which assigns a positive integer value to each element of P.

μ is the marking of the graph;

μ is a function from P to the non-negative numbers giving the marking of the net;

The marking is a vector

$\mu = (\mu_1, \mu_2, \dots \mu_n)$*, where* μ_i *is the* marking for the place p_i.

3. f(p) is the marking of the place p.

Marking is represented on the graph with tokens i.e., dots

Petri Nets elements

○	Place
▯	Transition
→ ←	Edge

Example

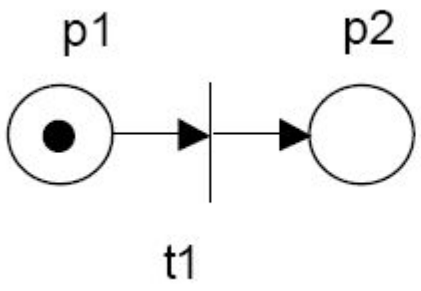
- □ This Petri net has:

2 places: p1, p2

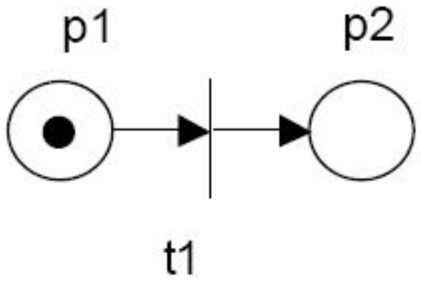
1 transition: t1

p1 has one token: f(p1) = 1

p2 has 0 tokens: f(p2) = 0

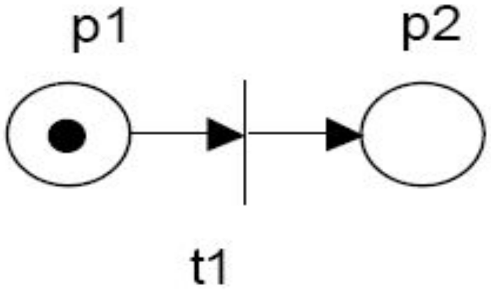


- Firing a Transition

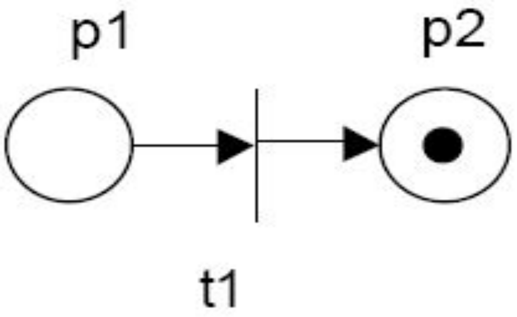


Example

- Petri net before t1 fires:



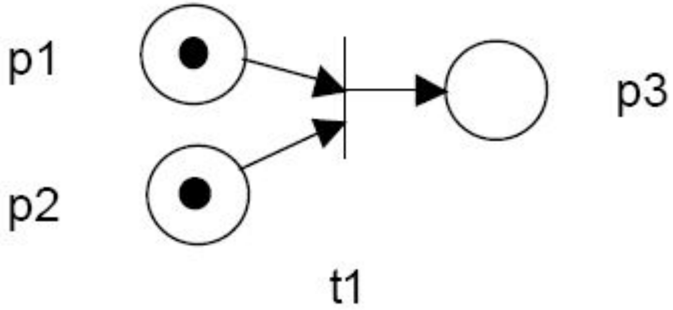
Petri net after t1 fires:



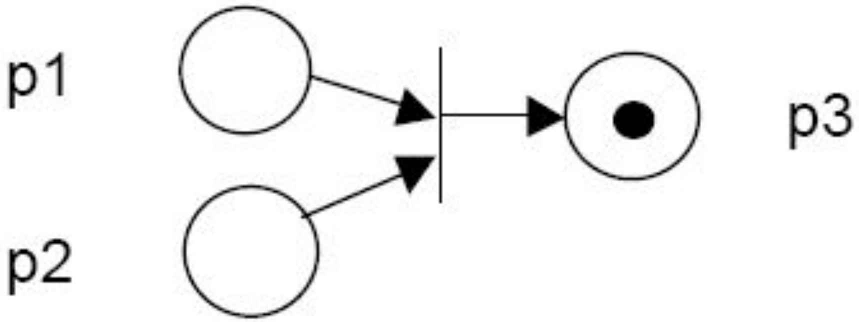
- A transition must be enabled before it fires
- There is a token in each pi that has an edge to the transition

Example

□ Petri net before t1 fires:



Petri net after t1 fires:



Only 1 token can be removed/added from a place when a transition fires

Other Types of Petri Nets

- Petri nets have been extended over the years in many directions including time, data, and hierarchy.

Time Extended Petri nets

- First developed in the mid 1970s

- For real systems it is often important to describe the *temporal behavior of the* system, i.e., we need to model durations and delays

Time Extended Petri nets

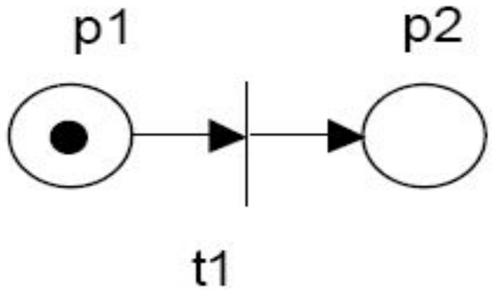
- There are 3 basic ways to introduce time into the Petri net. Time can be associated with:

  tokens

  places

  transition

Time Extended Petri nets



- The firing rules in this model are that the transition must fire as soon as the next place is empty, and firing a transition takes a fixed amount of time

Coloured Petri Nets

- Developed in the late 1970s by K. Jensen, “Coloured Petri nets and the invariant method”, Theoretical Computer Science, volume 14, 1981, pp. 317-336

- Tokens often represent objects (e.g. resources, goods, humans) in the modeled system

Coloured Petri Nets

□ To represent attributes of these objects, the Petri net model is extended with *coloured tokens*

each token has a value often referred to as `colour’

Hierarchical Petri Nets

□ Developed in the late 1980s

□ Specifications for real systems have a tendency to become large and complex

□ An abstraction mechanism, hierarchical structuring, is used to make constructing, reviewing, and modifying the model easier

Hierarchical Petri Nets

- The hierarchy construct is called a *subnet*
- A subnet is an aggregate of a number of places, transitions, and subsystems

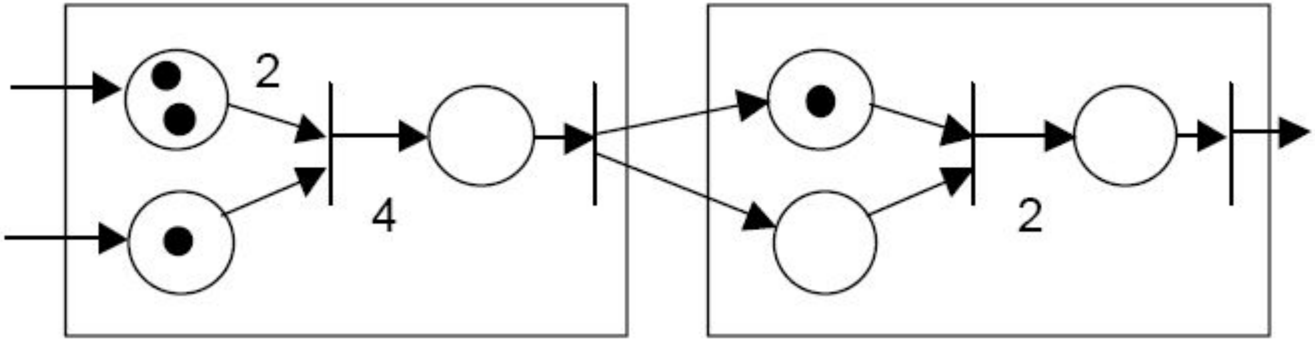
Hierarchical Petri Nets

- Such a construct can be used to structure large processes

- At one level we want to give a simple description of the process (without having to consider all the details). At another level we want to specify a more detailed behavior

Hierarchical Petri Nets

Each subnet is represented with a rectangular box that includes part of the Petri Net model



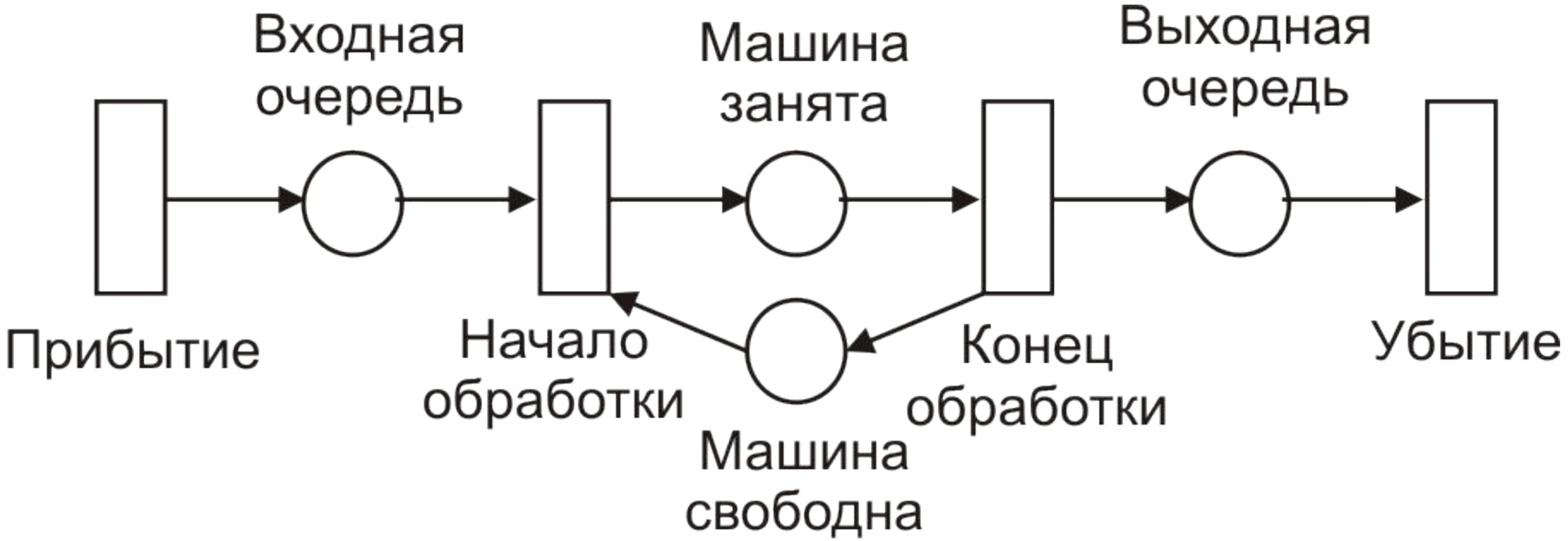
Properties of a Petri Net

- ***Terminate*** Does the Petri Net terminate?
- ***Immediately*** Reachable Is a state reachable when a transition fires?
- ***Reachable*** Is a state eventually reachable?
- ***Live*** In all states, is there at least one transition that can fire?
- ***Partial deadlock*** Is there a state in which at least one transition that can never fire?

Properties of a Petri Net

- ***Deadlock*** Is there a state in which none of the transitions can fire?
- ***Safe*** In all states, does each place contain at most one token?
- ***Bounded*** In all states, is there a limit to the number of tokens that can be in one place?
- ***Conservative*** Is the total number of tokens in the Petri Net constant?

Petri Net Example



Transitions are the events that changed object states in the real system

2. **Queuering systems**

Queuering system consists of one or more services processing generated entities. If service is busy (in the moment of the entity arrival) then entity takes a place in the queue that is associated with this service.

Queuering system components

Generator	generates arriving entities in the model and time of arrival
Service	Service is a processing unit associated with the queue. It has a definite time of entity processing
Queue	It is a rule according to which each service chooses an entity for processing (FIFO, LIFO, priority queue)

3. **Combined models**

Usually combined model is a complex model that is based on two or more mathematical formalisms

Models created by means of Arena 7.0 are combined models (colour petri net and queuering system)

Simulation tool Arena 7.0

Arena 7.0 was developed by Systems Modeling (Rockwell Software)

Arena 7.0 allows to

1. Formalize and visualize dynamics of complex processes and systems

2. Analyze work flow

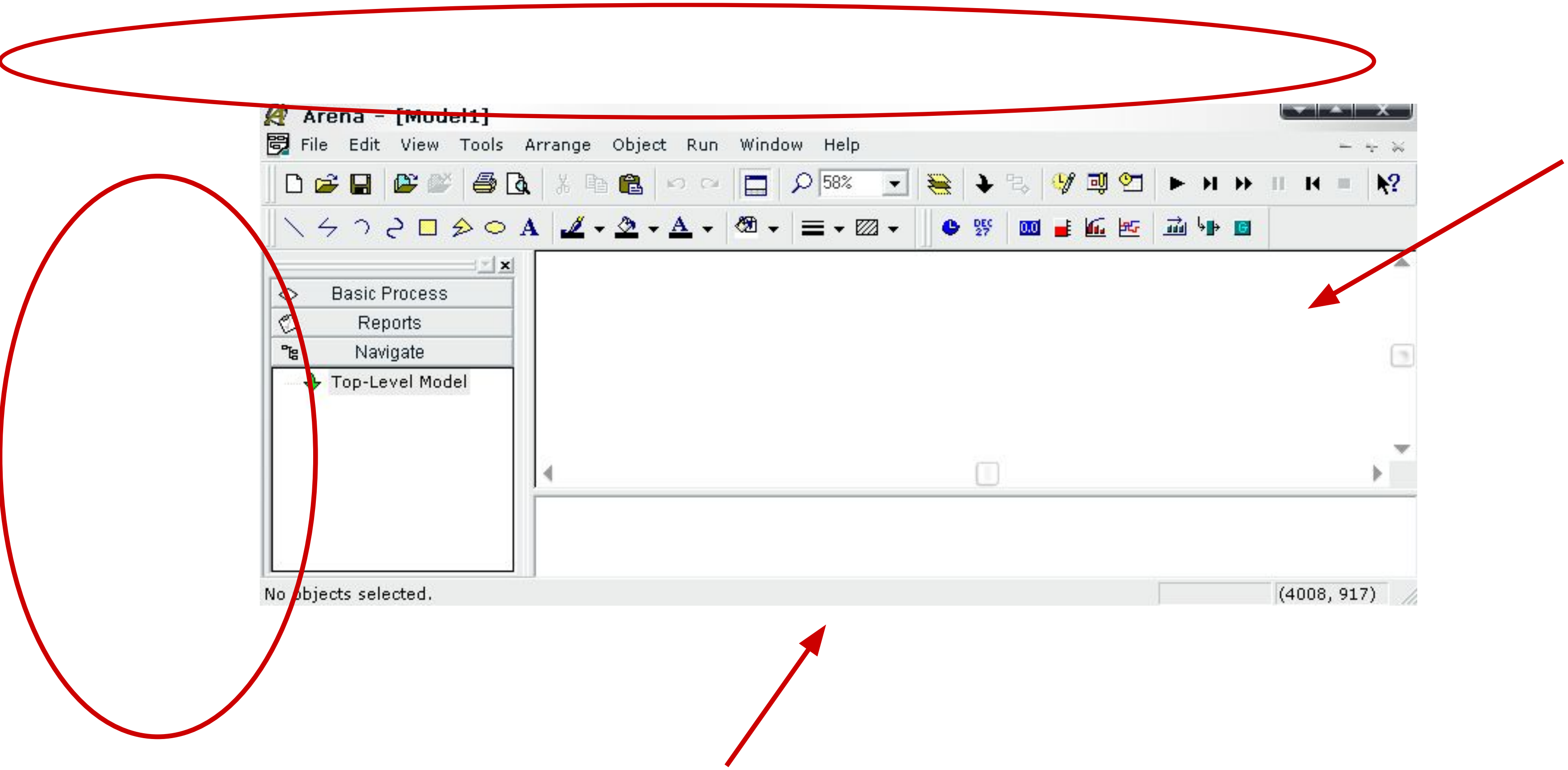
3. Optimize and analyze business processes

Arena 7.0 allows to

4. Find an optimal recourses distribution (humans, equipments, finances)

5. Forecast system behavior

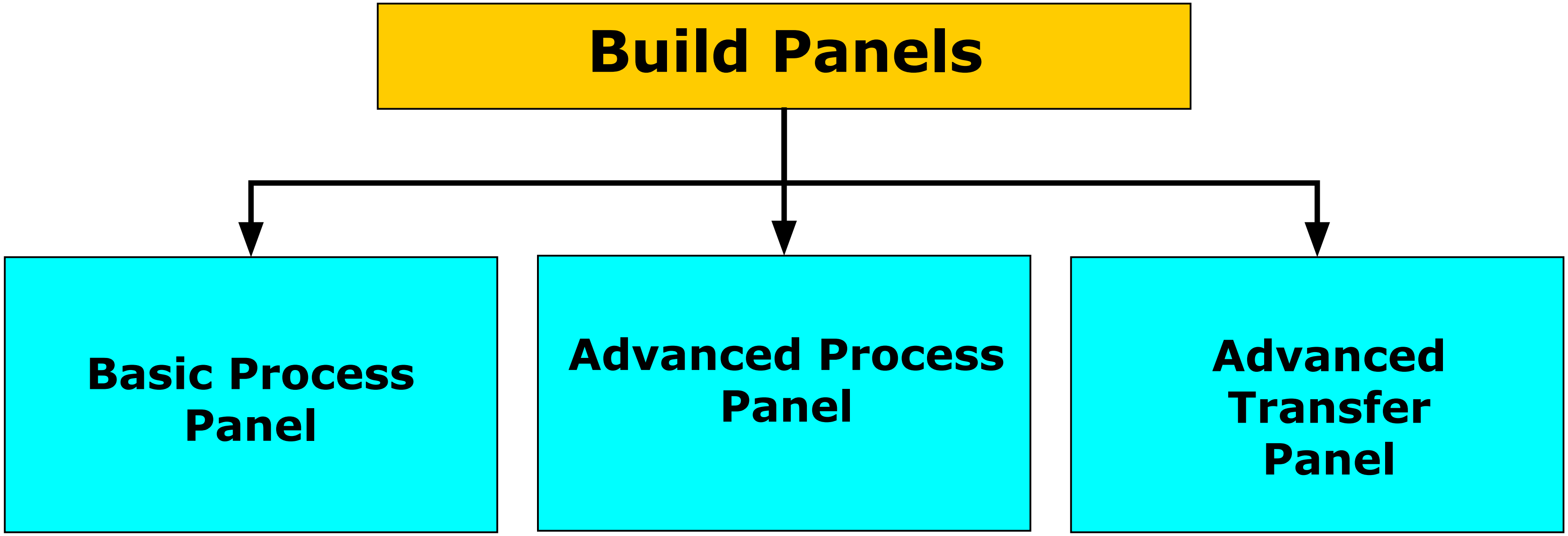
Main window in Arena 7.0



No objects selected.

(4008, 917)

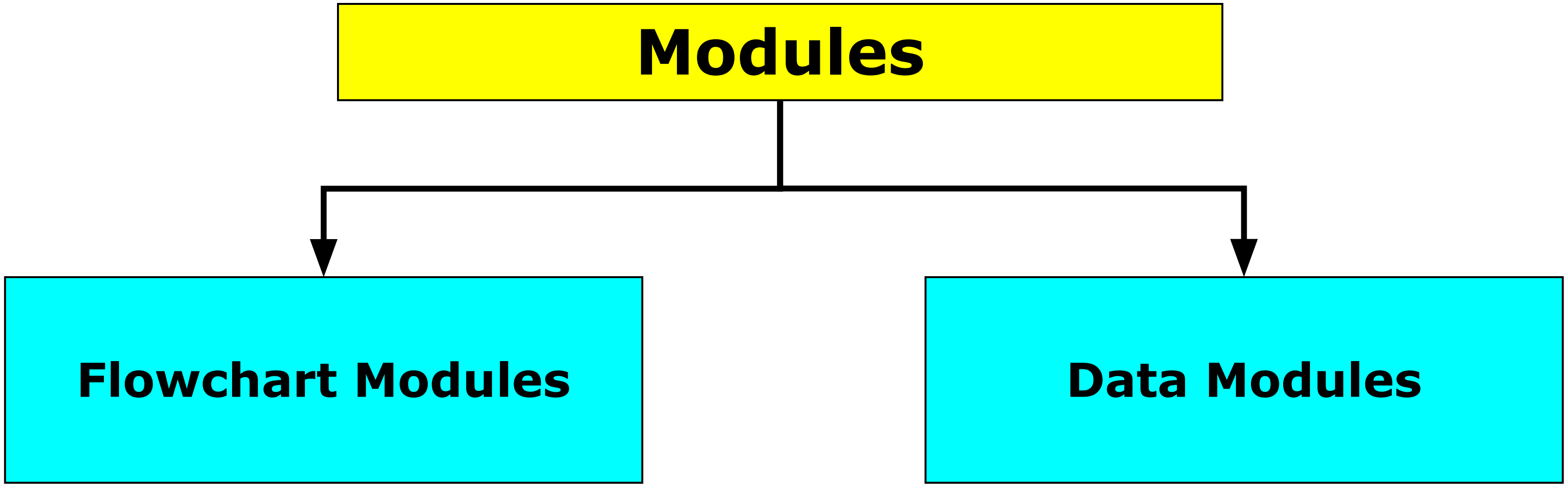
Build Panels


Build Panels

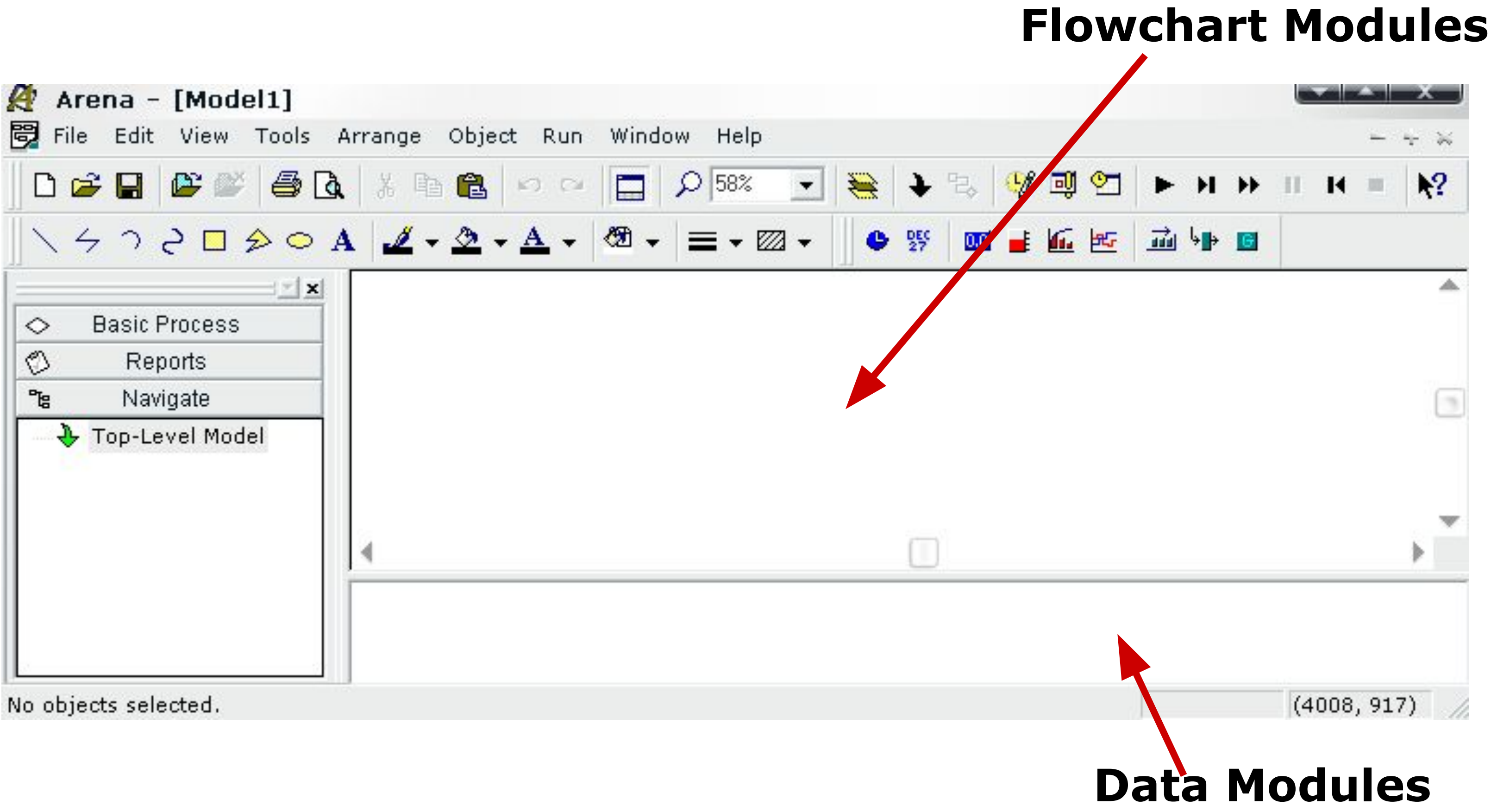
Basic Process Panel

Advanced Process Panel

Advanced Transfer Panel


Modules
Flowchart Modules
Data Modules

Main window in Arena 7.0



1. Basic Process Panel

1.1 Flowchart Modules

1.1.1 Create

1.1.2 Process

1.1.3 Decide

1.1.4 Batch

1.1.5 Separate

1.1.6 Assign

1.1.7 Record

1.1.8 Dispose

1. Basic Process Panel

1.2 Data Modules

1.2.1 Entity

1.2.2 Queue

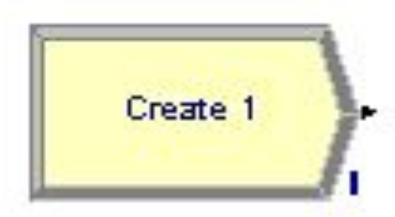
1.2.3 Resource

1.2.4 Schedule

1.2.5 Set

1.2.6 Variable

1.1.1 Create



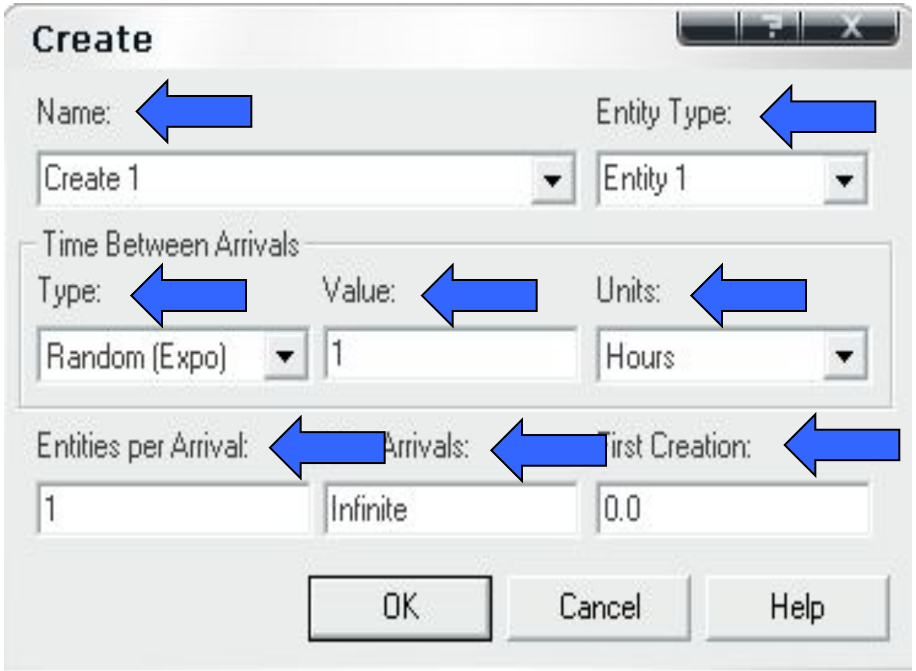
Create module allows to generate entities in the model

Entity is an element, that will be processed in a model (client requests, details, claims and others)

Application of Create module

- □ Document arrival
- □ Client coming
- □ Starting point of production in technological process

Create module parameters



Name: Create 1

Entity Type: Entity 1

Time Between Arrivals

Type: Random (Expo)

Value: 1

Units: Hours

Entities per Arrival: 1

Arrivals: Infinite

irst Creation: 0.0

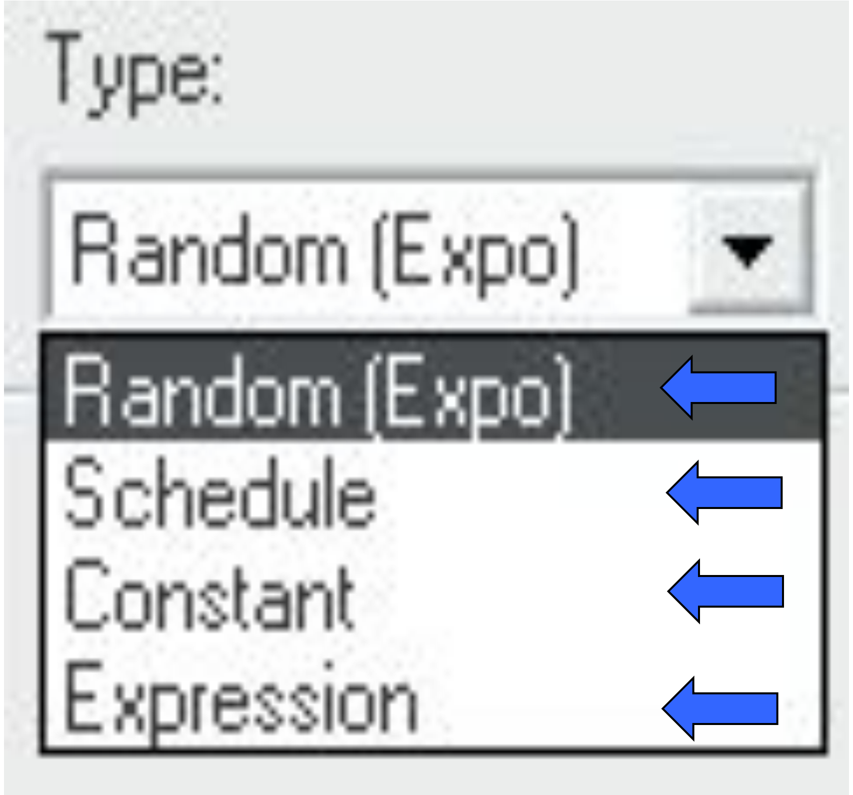
- □ Name
- □ Entity Type

Time between arrivals

- □ Type
- □ Value
- □ Units
- □ Entities per Arrival

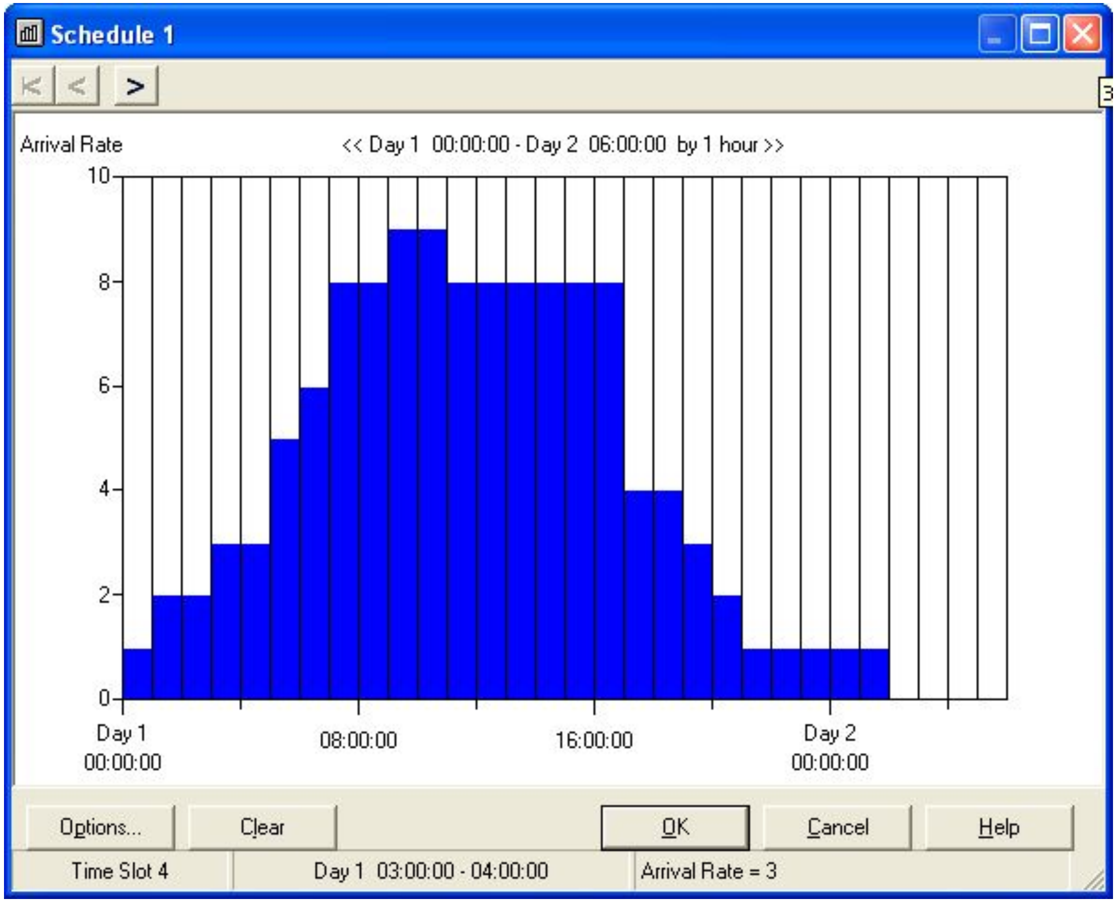
- □ Max Arrivals
- □ First Creation

Create module parameter: Type



- □ **Random**
- □ **Schedule**
- □ **Constant**
- □ **Expression**

Entity Arrival by Schedule

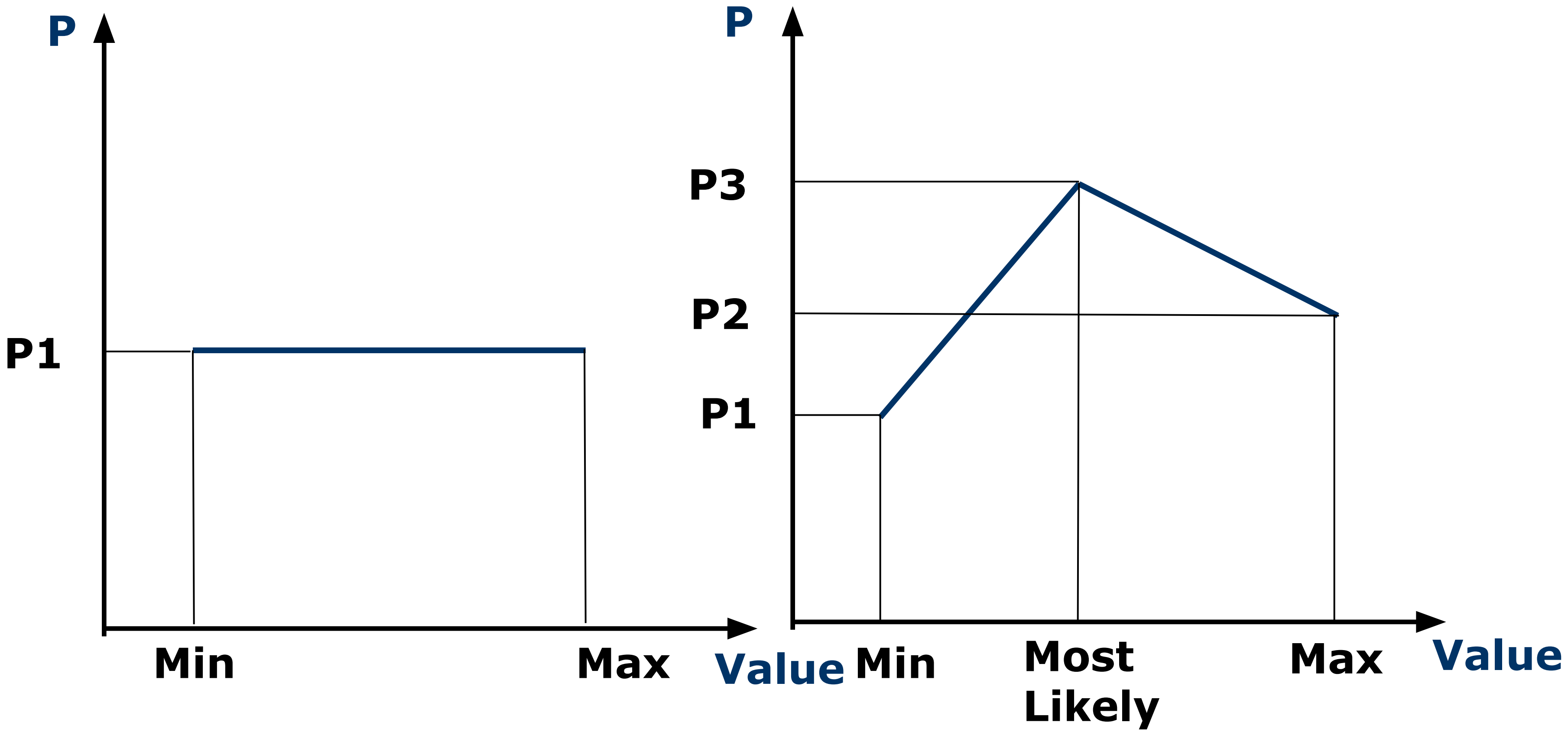


Time Slot 4 | Day 1 03:00:00 - 04:00:00 | Arrival Rate = 3

Typical probability distributions

- □ Normal : Mean, StdDev
- □ Exponential : Mean
- □ Uniform : Min, Max
- □ Poisson : Mean
- □ Gamma : Beta, Alpha
- □ Beta : Beta, Alpha
- □ Triangular : Min, Mode, Max

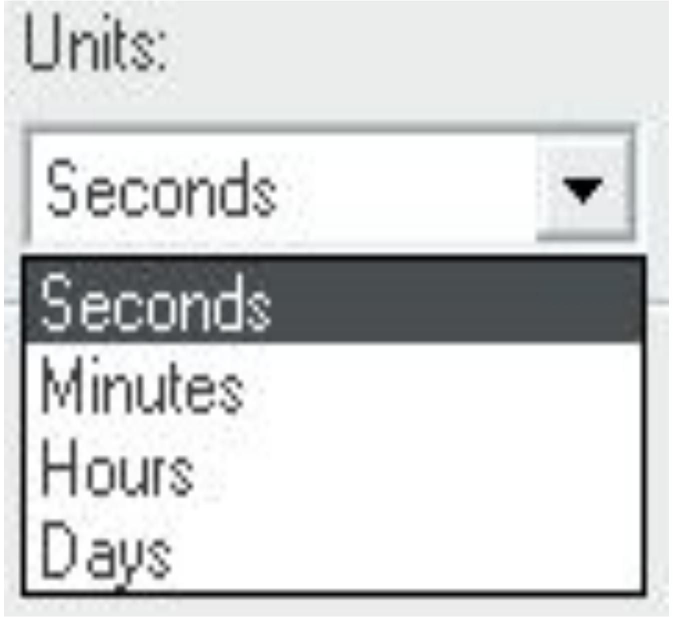
Uniform and Triangular distributions



Uniform distribution

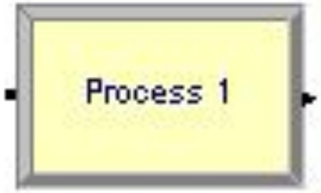
Triangular distribution

Create module parameter: Units


Units: Seconds

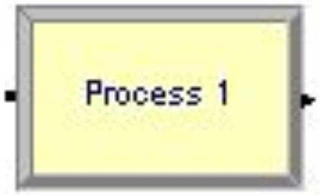
- Seconds
- Minutes
- Hours
- Days

1.1.2 Process



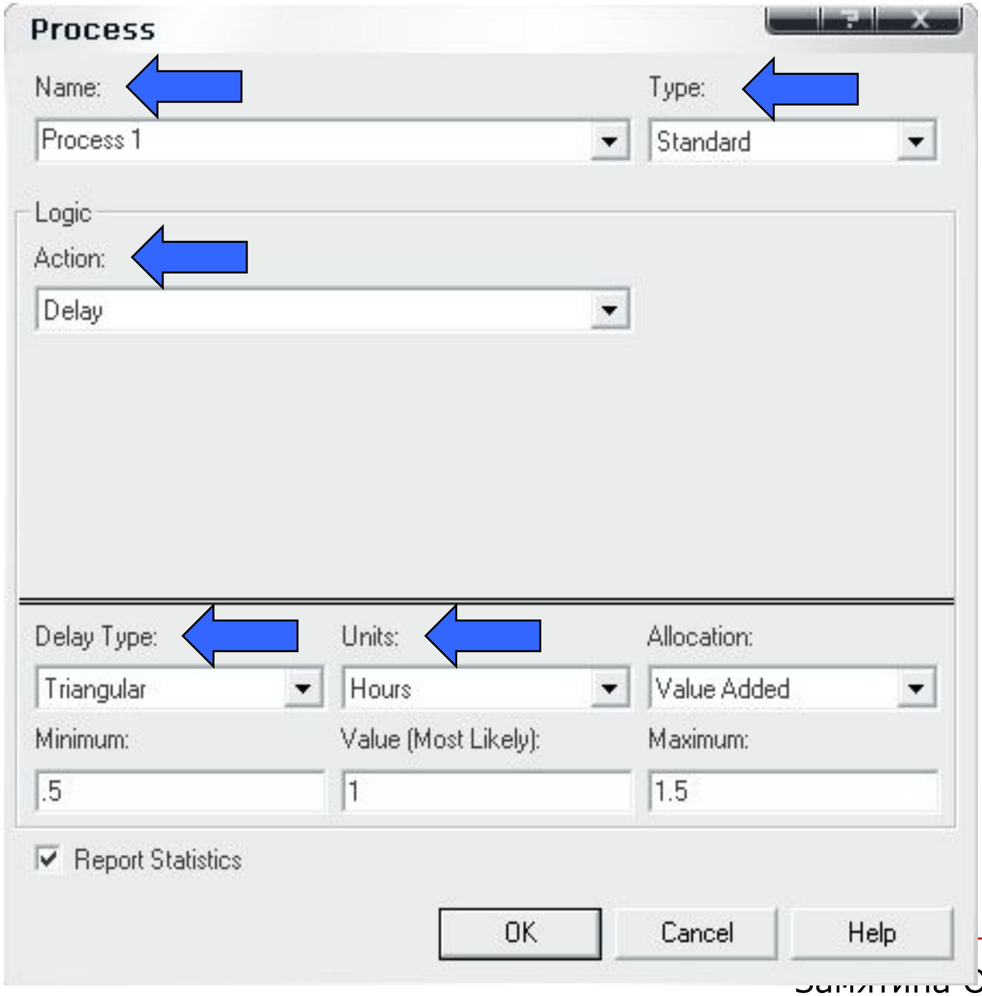
Process is the main module. It intends for entity processing

Application of Process module



- □ Document checking
- □ Order performing
- □ Client service
- □ Part cutting

Process module parameters



Process

Name: Process 1

Type: Standard

Logic

Action: Delay

Delay Type: Triangular

Units: Hours

Allocation: Value Added

Minimum: .5

Value (Most Likely): 1

Maximum: 1.5

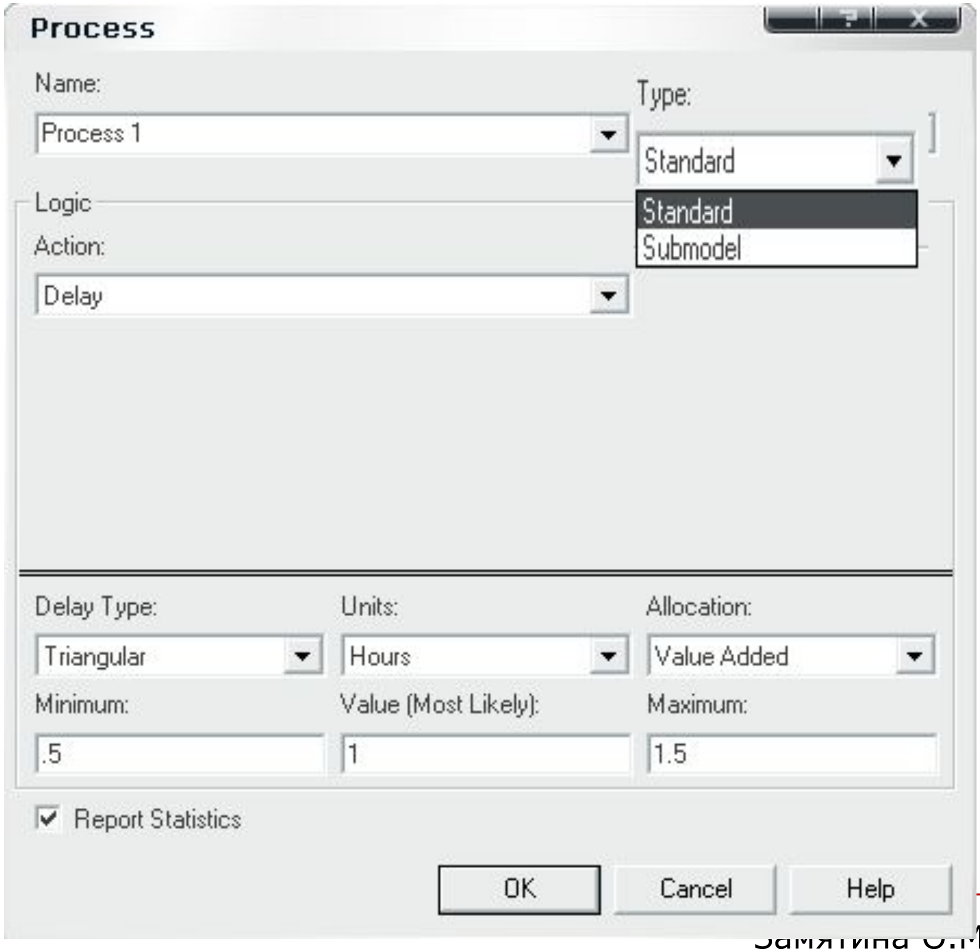
☑ Report Statistics

- Name
- Type

Process logic

- Action
- Delay Type
- Units

Process module parameter: Type



Process

Name: Process 1

Type: Standard

Logic

Action: Delay

Delay Type: Triangular

Units: Hours

Allocation: Value Added

Minimum: .5

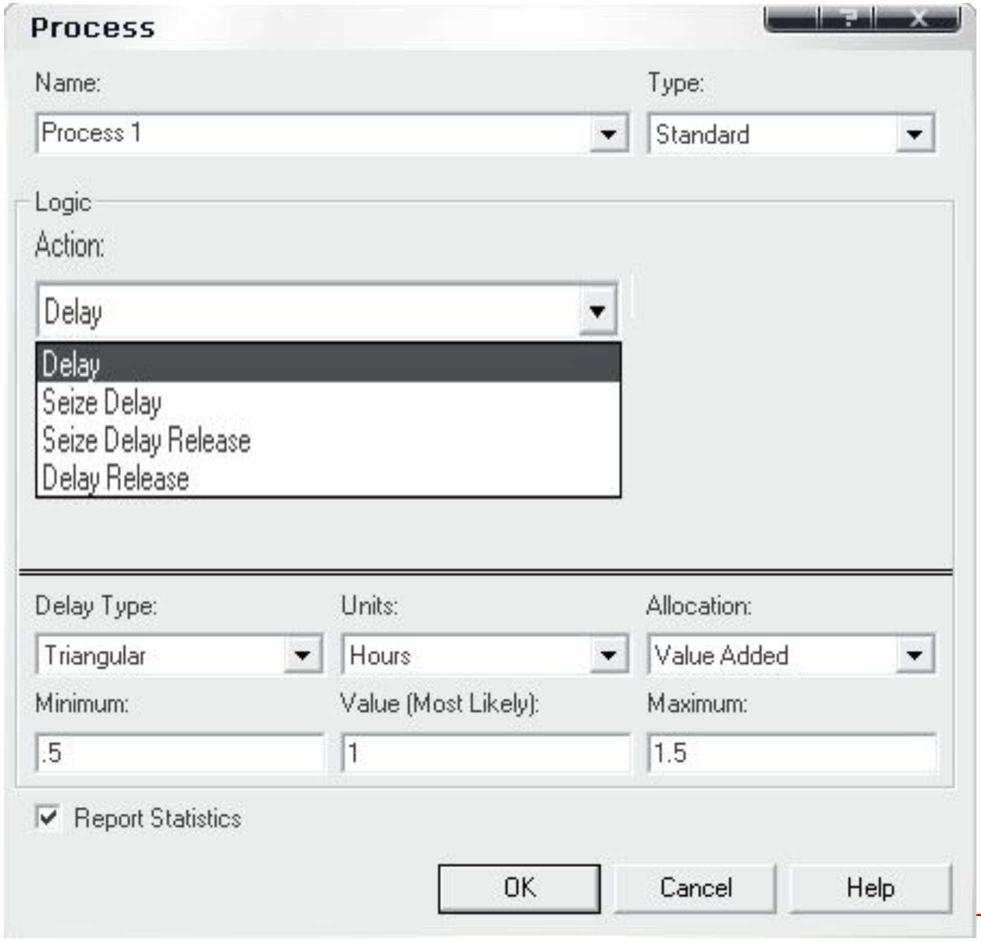
Value (Most Likely): 1

Maximum: 1.5

☑ Report Statistics

- □ Standard
- □ Submodel

Process module parameter: Action



Process

Name: Process 1

Type: Standard

Logic

Action: Delay

Delay
Seize Delay
Seize Delay Release
Delay Release

Delay Type: Triangular

Units: Hours

Allocation: Value Added

Minimum: .5

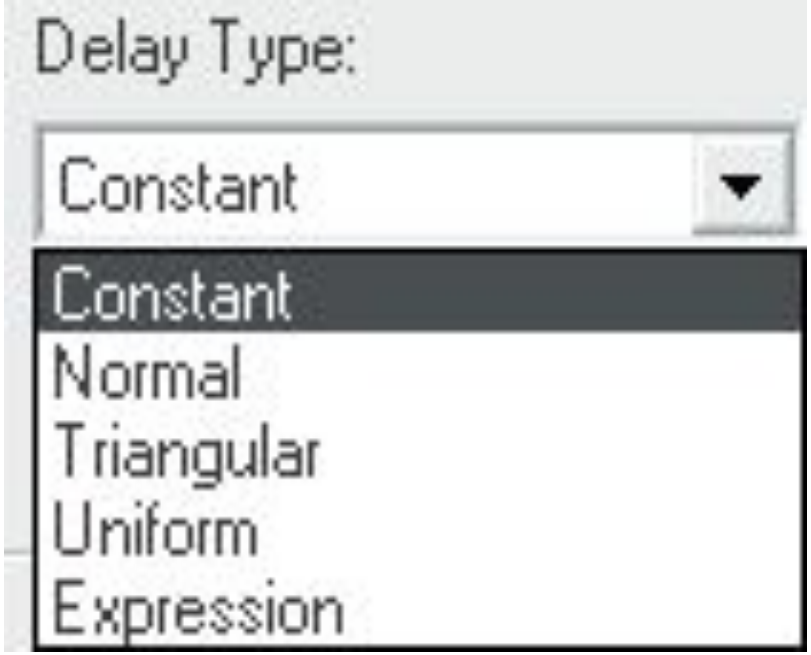
Value (Most Likely): 1

Maximum: 1.5

☑ Report Statistics

- Delay
- Seize Delay
- Seize Delay Release
- Delay Release

Process module parameter: Delay Type

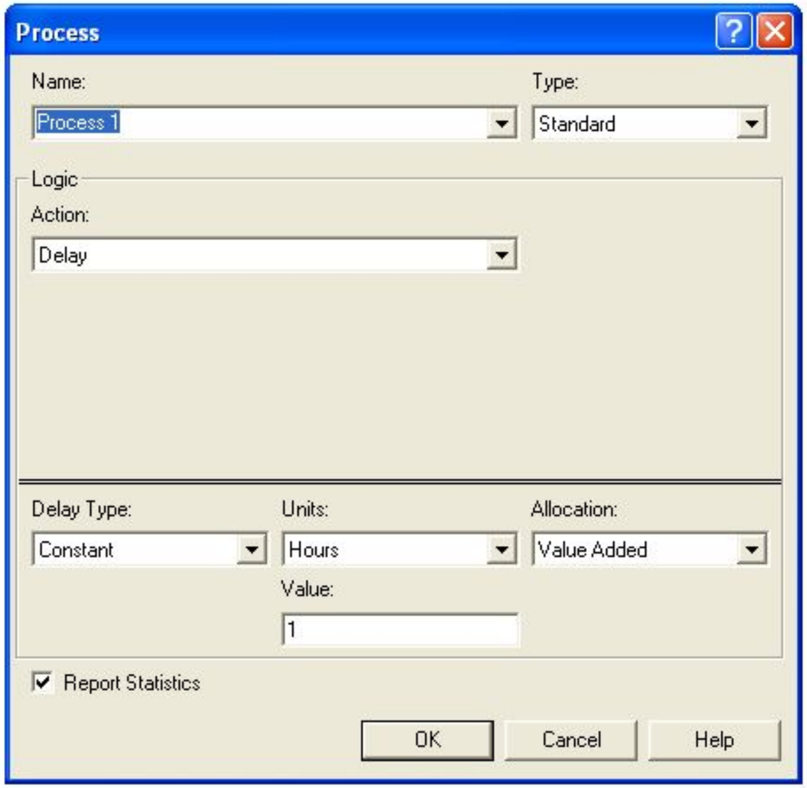


Delay Type:

Constant

- Constant
- Normal
- Triangular
- Uniform
- Expression

Process module examples



Name: Process 1

Type: Standard

Action: Delay

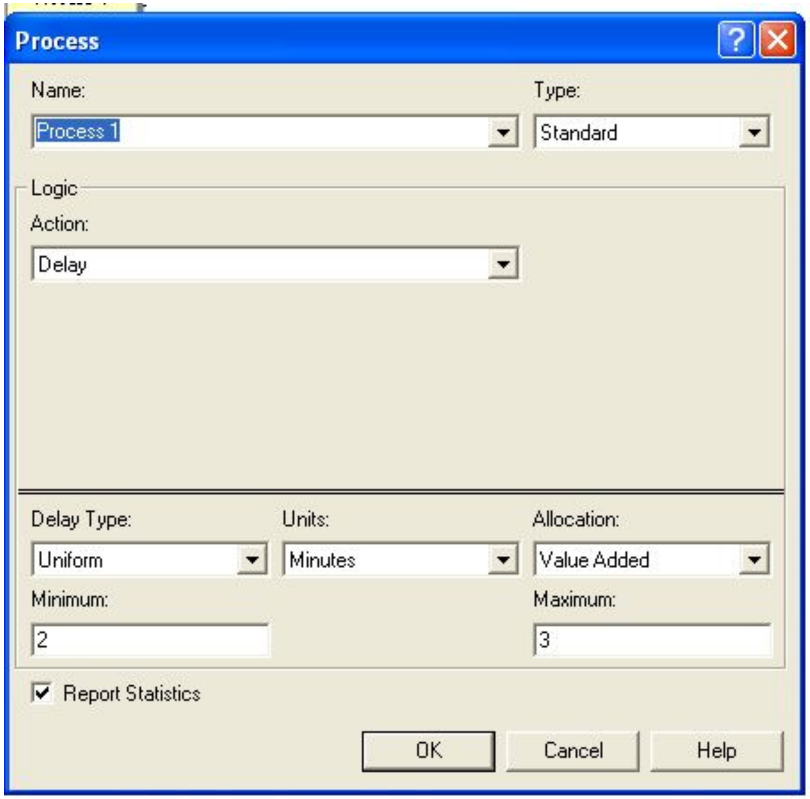
Delay Type: Constant

Units: Hours

Allocation: Value Added

Value: 1

☑ Report Statistics



Name: Process 1

Type: Standard

Action: Delay

Delay Type: Uniform

Units: Minutes

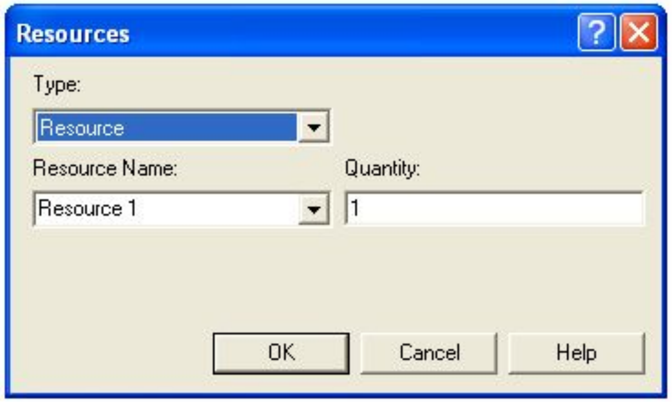
Allocation: Value Added

Minimum: 2

Maximum: 3

☑ Report Statistics

Process module examples

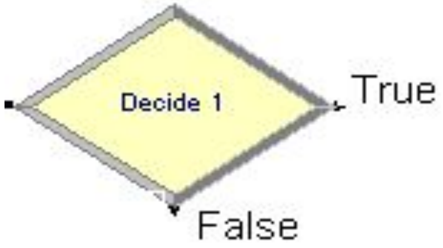


Type: Resource

Resource Name: Resource 1

Quantity: 1

1.1.3 Decide



Decide allows to set process logic

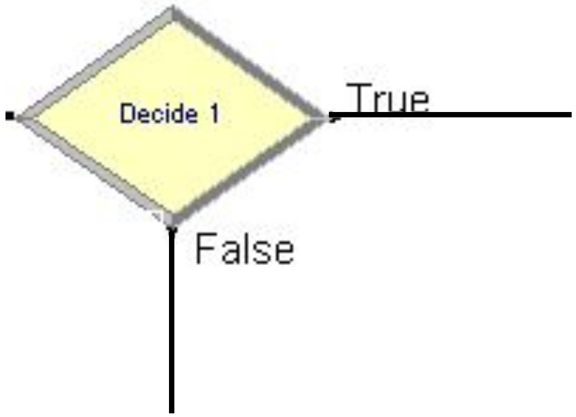
If condition defined in the Decide module is right when entity goes in the direction True, otherwise it goes in the direction False.

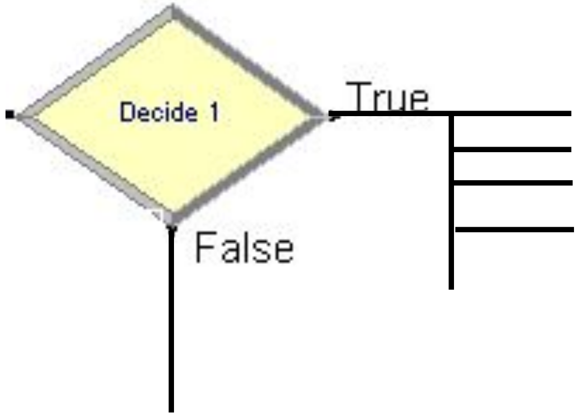
1.1.3 Decide

Decide module can be:

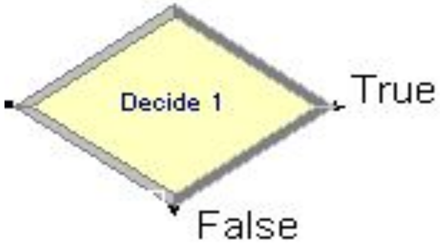
2-way

N-way



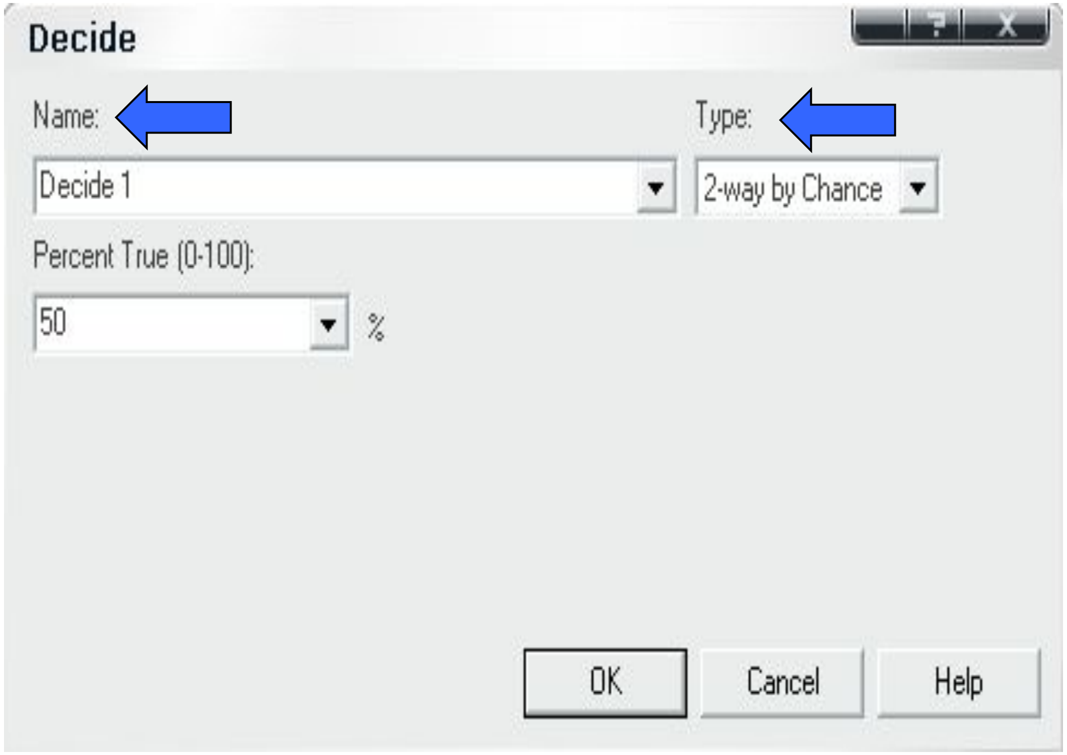


Application of Decide module



- □ Sorting (pedestrians at a bus stop, drivers at a parking)
- □ Sorting according to logic properties (parts from 1 to 19 go to Ivanov, other parts – to Petrov)
- □ Redirection unfinished or defective parts

Decide module parameters



Decide

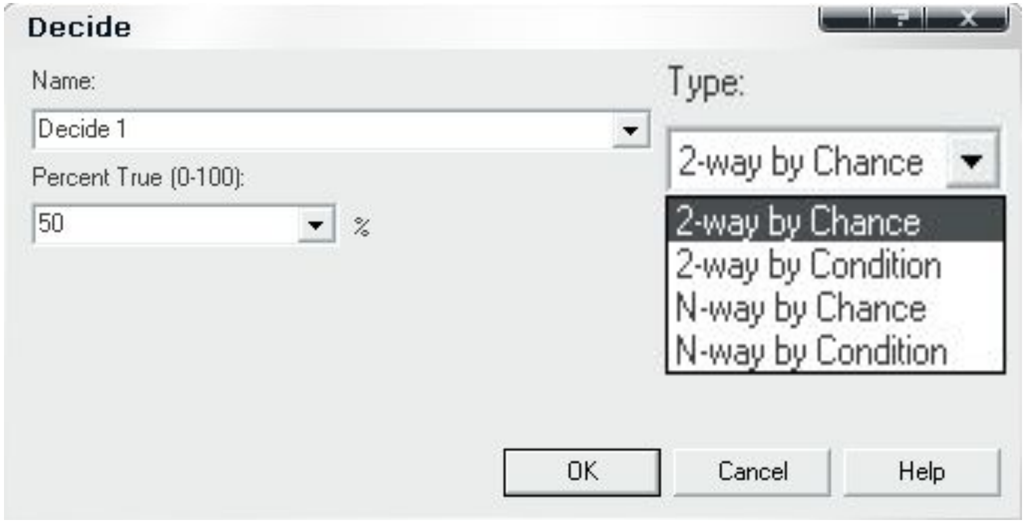
Name: Decide 1

Type: 2-way by Chance

Percent True (0-100): 50 %

- □ Name
- □ Type

Decide module parameter: Type



Decide

Name: Decide 1

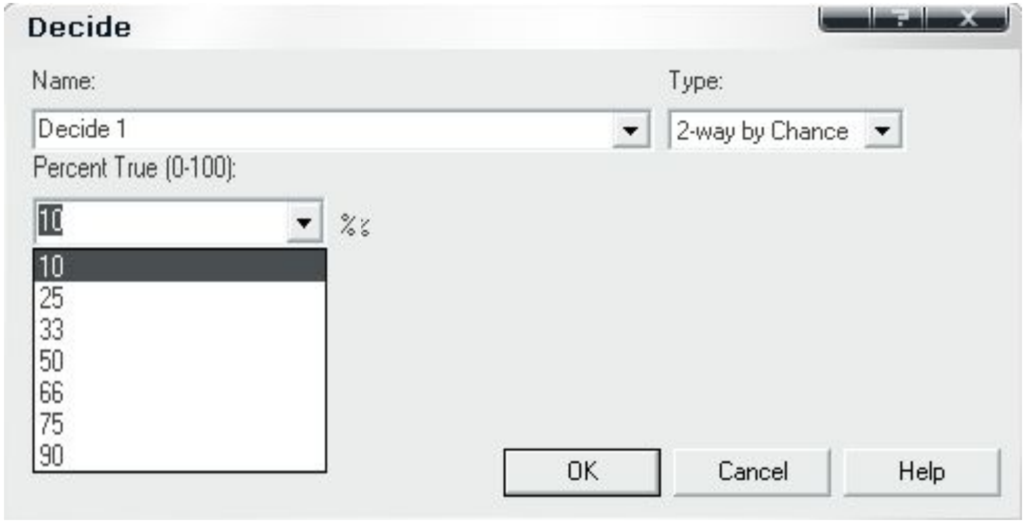
Percent True (0-100): 50 %

Type: 2-way by Chance

2-way by Chance
2-way by Condition
N-way by Chance
N-way by Condition

- 2-way by Chance
- 2-way by Condition
- N-way by Chance
- N-way by Condition

2-way by Chance



Decide

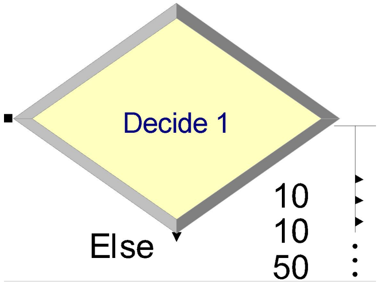
Name: Decide 1

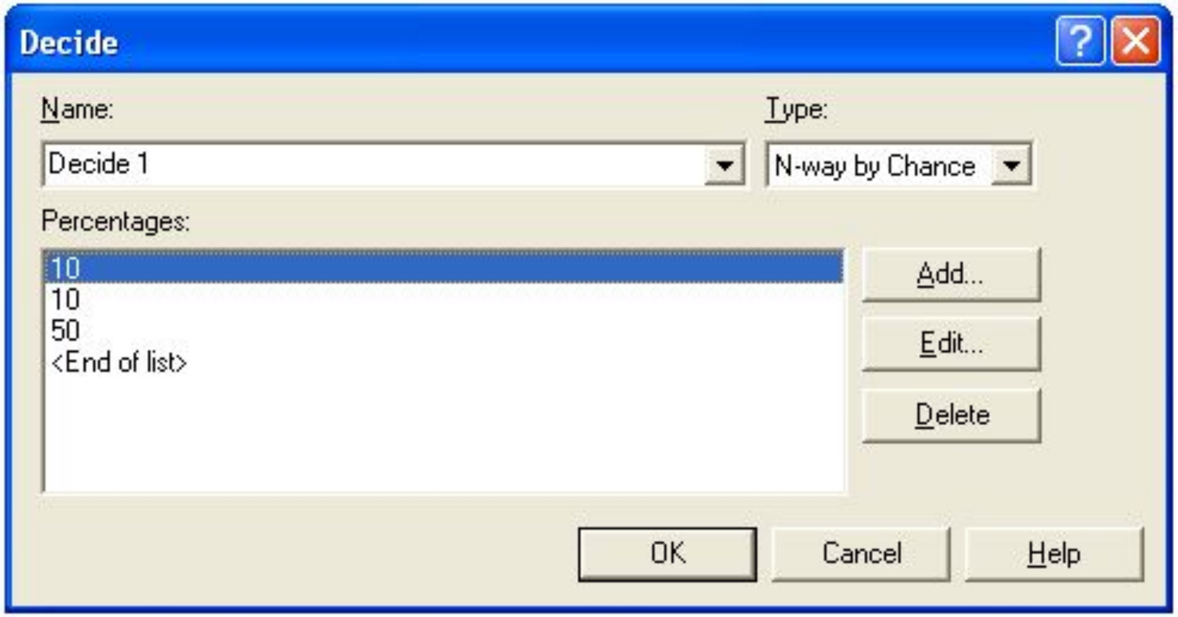
Type: 2-way by Chance

Percent True (0-100): 10 %

- □ 10
- □ 25
- □ 33
- □ 50
- □ 66
- □ 75
- □ 90

N-way by Chance



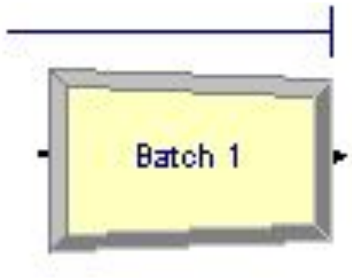

Decide

Name: Decide 1

Type: N-way by Chance

Percentages:
10
10
50
<End of list>

1.1.4 Batch

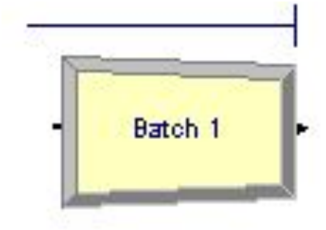


Batch module allows to create groups in a model

1.1.4 Batch

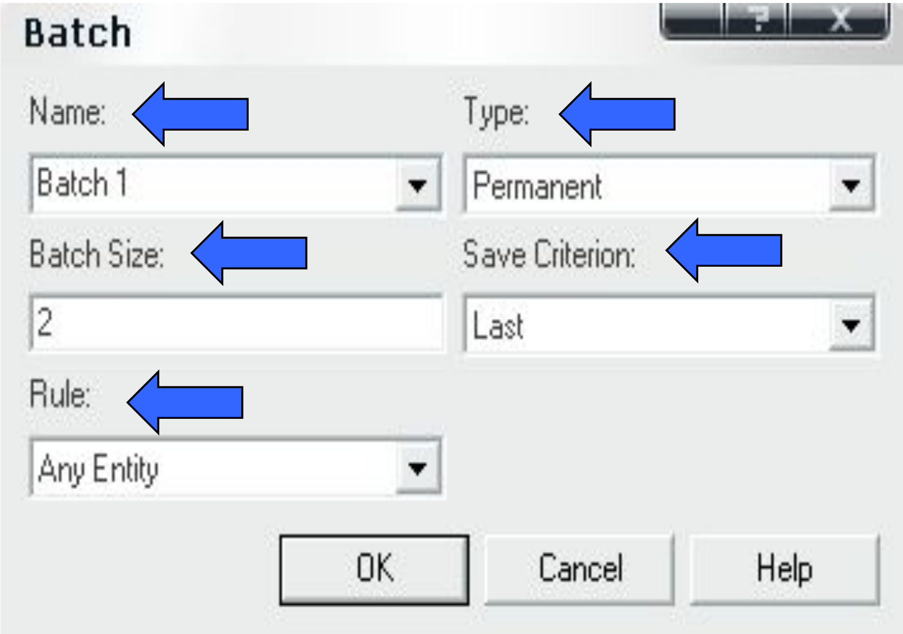
Entities arrive to Batch module and take a place in a queue. Entities are kept in the queue until its number equals the batch parameter. When a required number of entities is collected in a queue a new entity is created and is forwarded for further processing

Application of Batch module



- Collect necessary number of parts/data for their processing
- Collect earlier divided copies of one set
- Associate a patient and his medical history before doctor’s appointment

Batch module parameters



Batch

Name: Batch 1

Type: Permanent

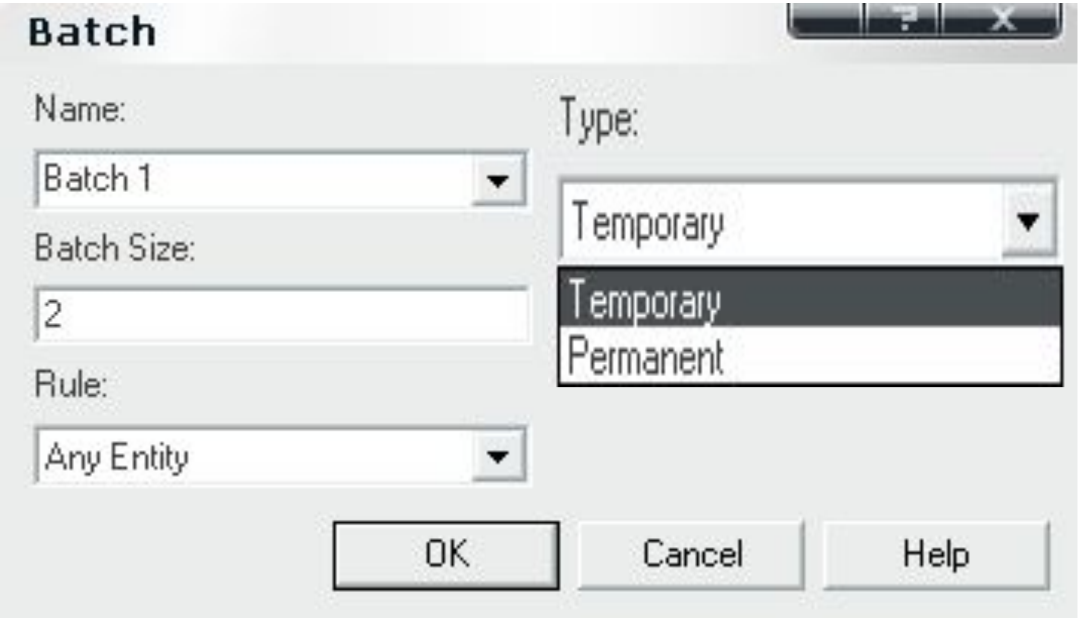
Batch Size: 2

Save Criterion: Last

Rule: Any Entity

- Name
- Type
- Batch Size
- Save Criterion
- Rule

Batch module parameter: Type



Batch

Name: Batch 1

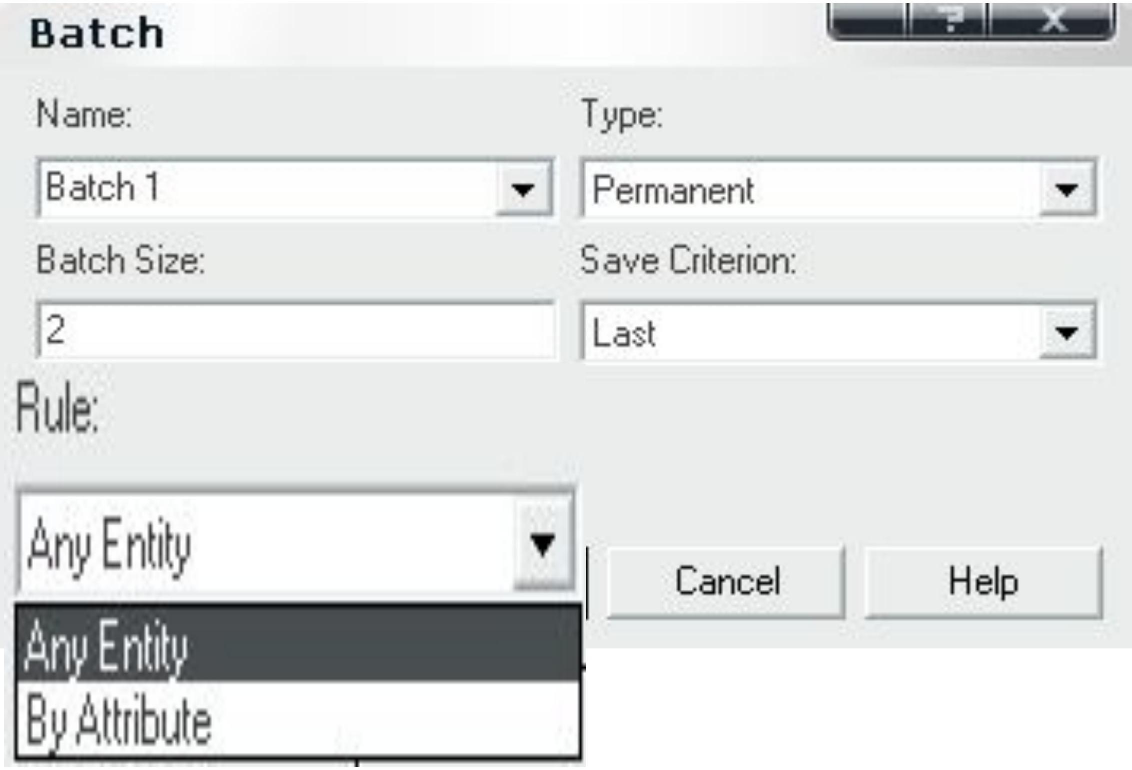
Batch Size: 2

Rule: Any Entity

Type: Temporary

- □ Temporary
- □ Permanent

Batch module parameter: Rule



Batch

Name: Batch 1

Type: Permanent

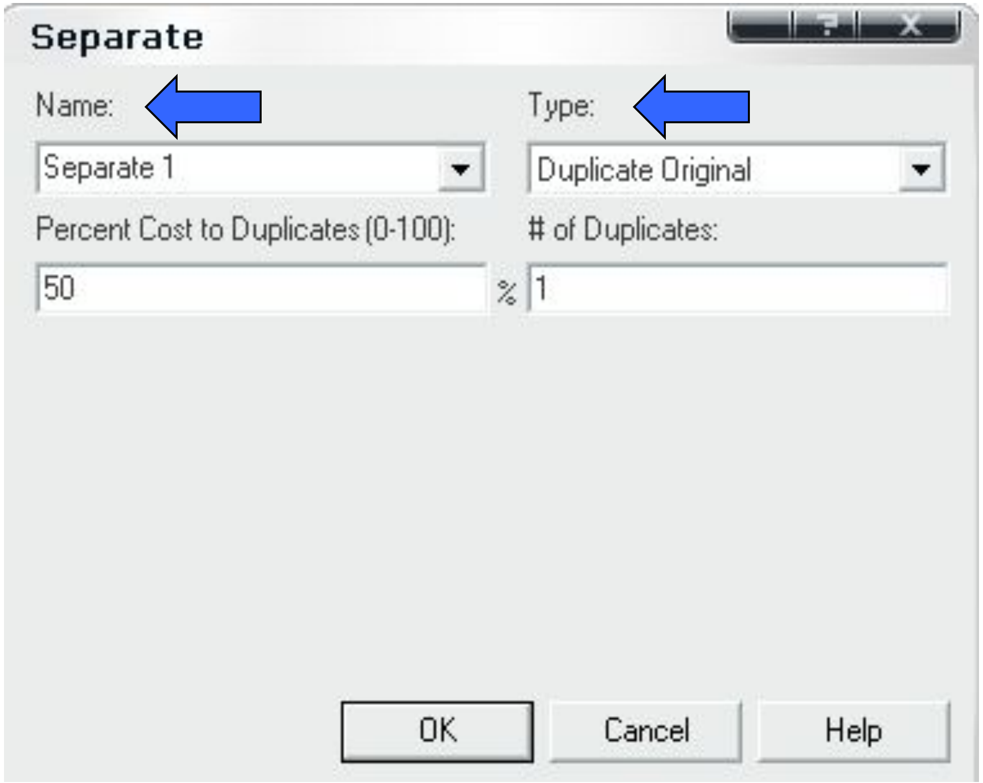
Batch Size: 2

Save Criterion: Last

Rule: Any Entity

- □ Any Entity
- □ By Attribute

1.1.5 Separate



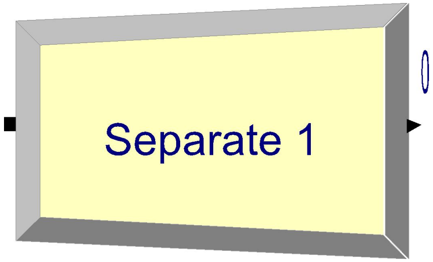
Separate

Name: Separate 1

Type: Duplicate Original

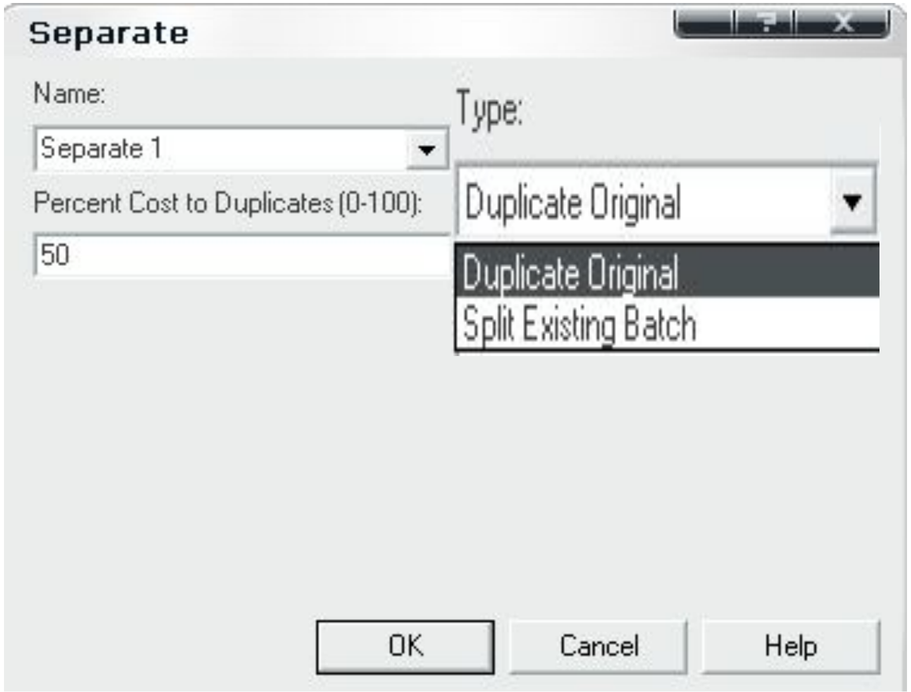
Percent Cost to Duplicates (0-100): 50 %

# of Duplicates: 1



- □ Name
- □ Type

Separate module parameter: Type



Separate

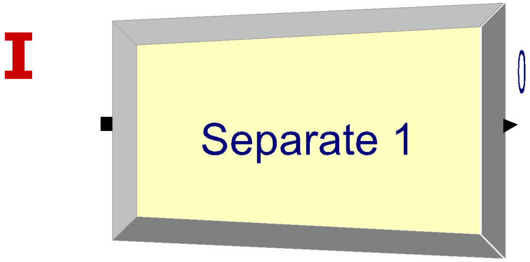
Name: Separate 1

Percent Cost to Duplicates (0-100): 50

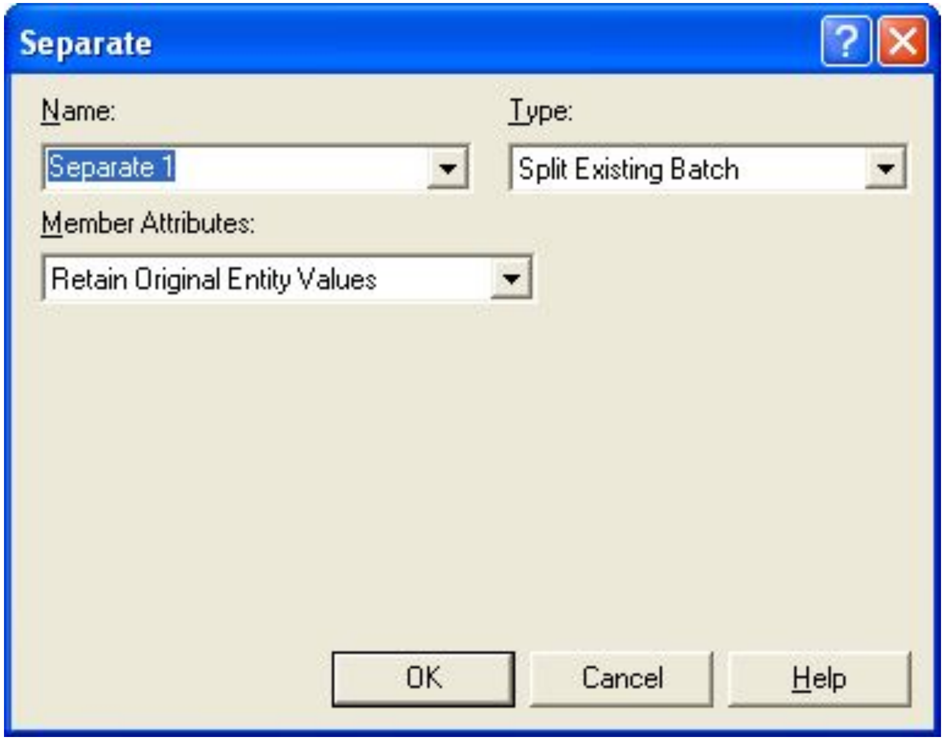
Type: Duplicate Original

- □ Duplicate Original
- □ Split Existing Batch

1.1.5 Separate



Separate module allows to divide earlier batched entities (on condition that batch type is temporary)



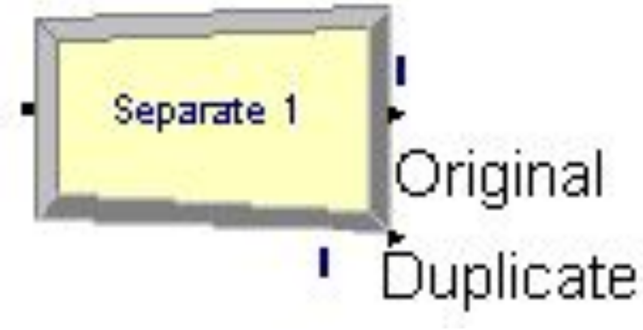
Name: Separate 1

Type: Split Existing Batch

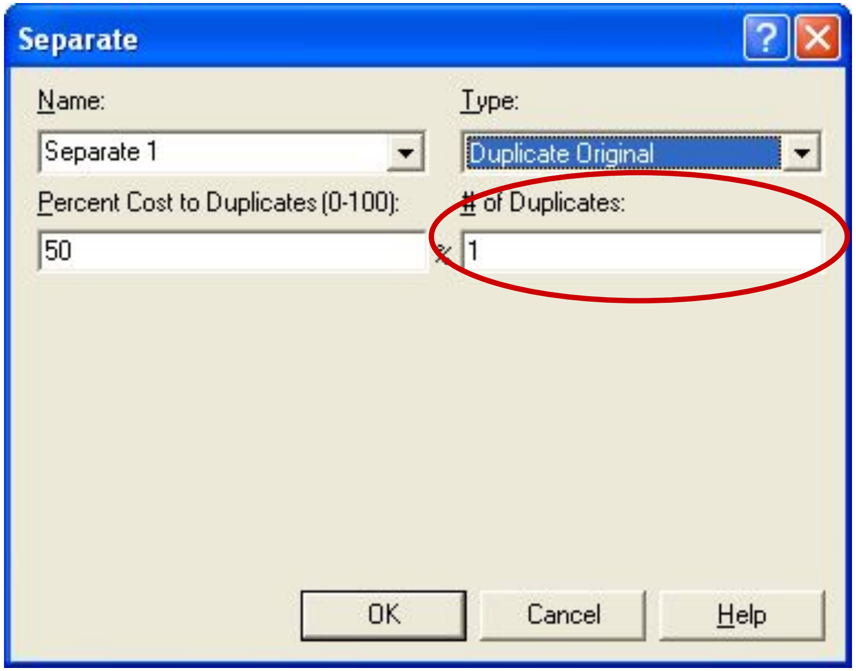
Member Attributes: Retain Original Entity Values

1.1.5 Separate

II



Separate module is used to make copies of arriving entities



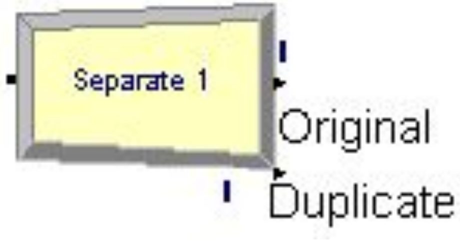
Name: Separate 1

Type: Duplicate Original

Percent Cost to Duplicates (0-100): 50 %

# of Duplicates: 1

Application of Separate module



- □ Separation of earlier batched entities
- □ Parallel processing of documents (invoices)

1.1.6 Assign

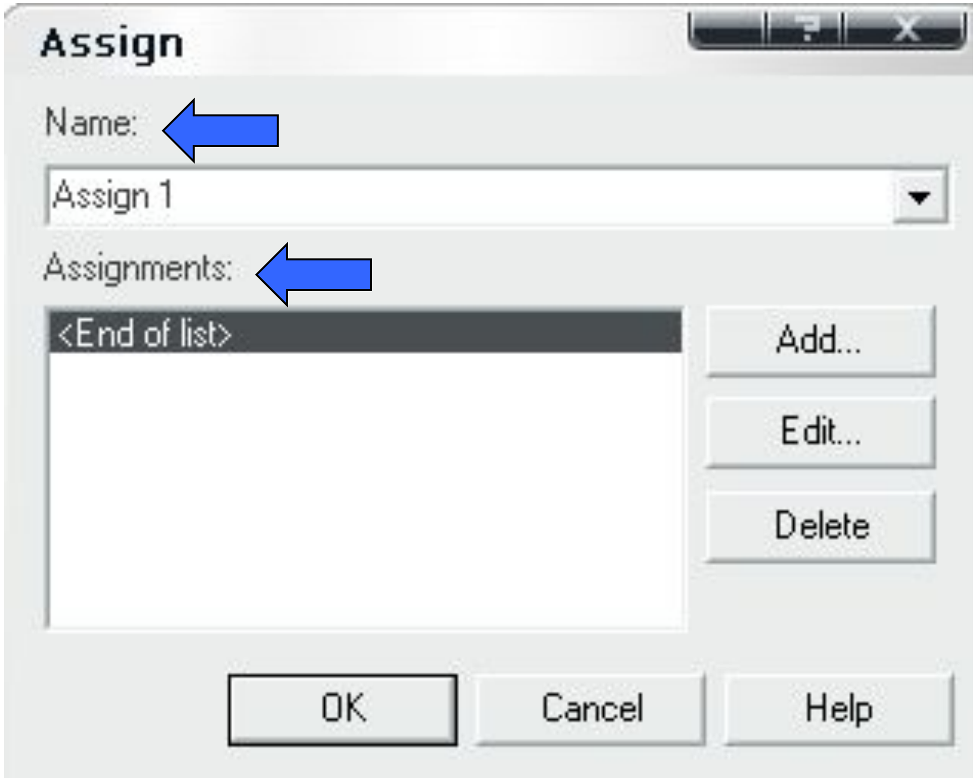
Assign module allows to set new value of attributes (entity’s type, entity’s picture) variables and so on

In the one *Assign* module you can made any number of assignments

Application of *Assign* module

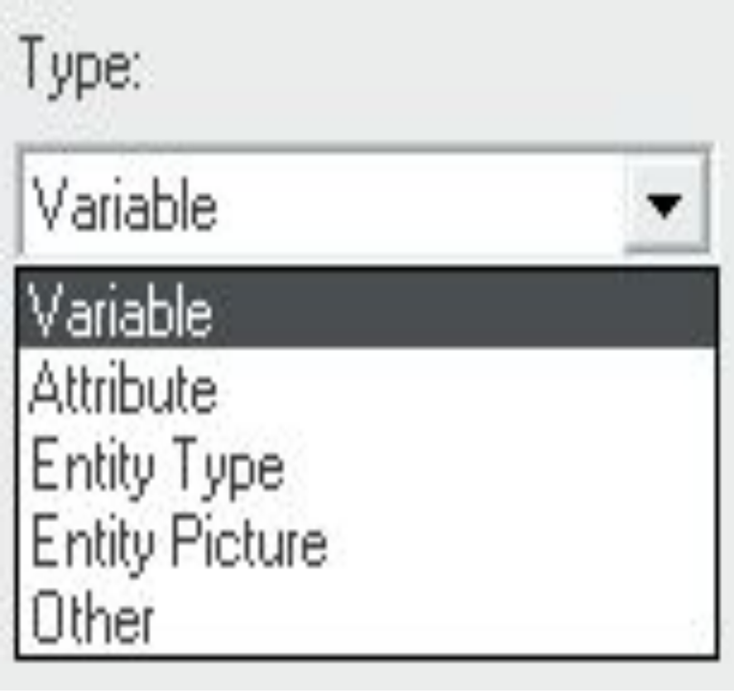
- □ Identification of entity number
- □ Changing of animation picture
- □ Setting of new value of variable

Assign module parameters



- □ Name
- □ Assignments

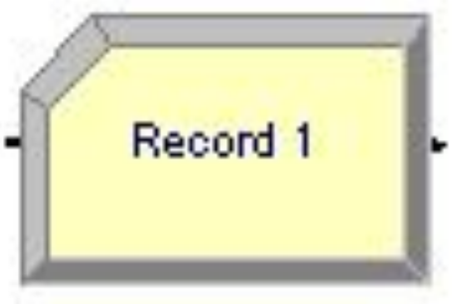
Assign module parameters: Assignments (Type)


Type:

Variable

- □ Variable
- □ Attribute
- □ Entity Type
- □ Entity Picture
- □ Other

1.1.7 Record

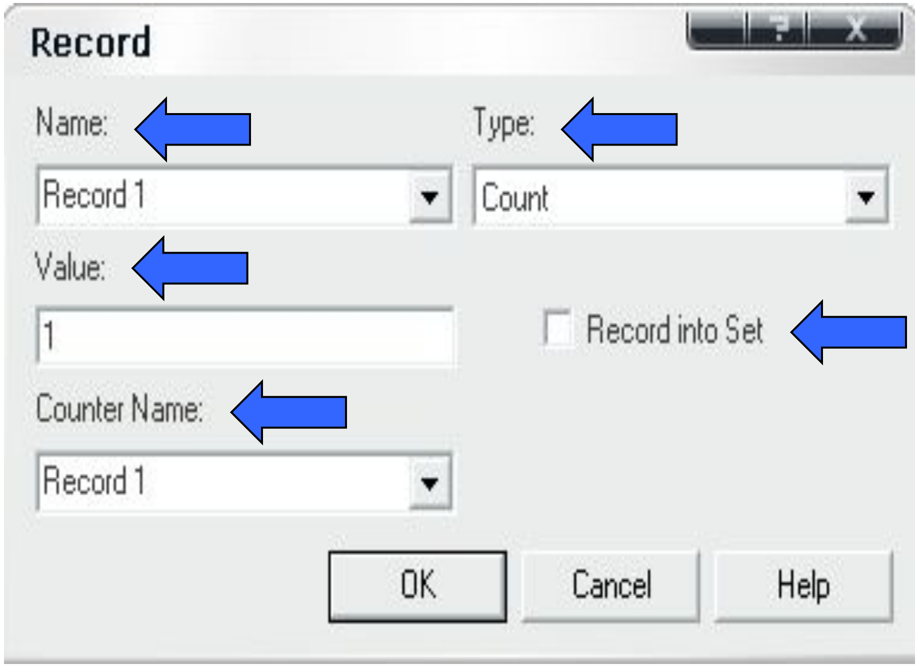


Record module intend for specific statistic data collection. It needs when it is lacking in standart reports

Application of Record module

- □ To count a number of request which were done with delay
- □ To count an amount of work which was done for time item

Record module parameter



Record

Name: Record 1

Type: Count

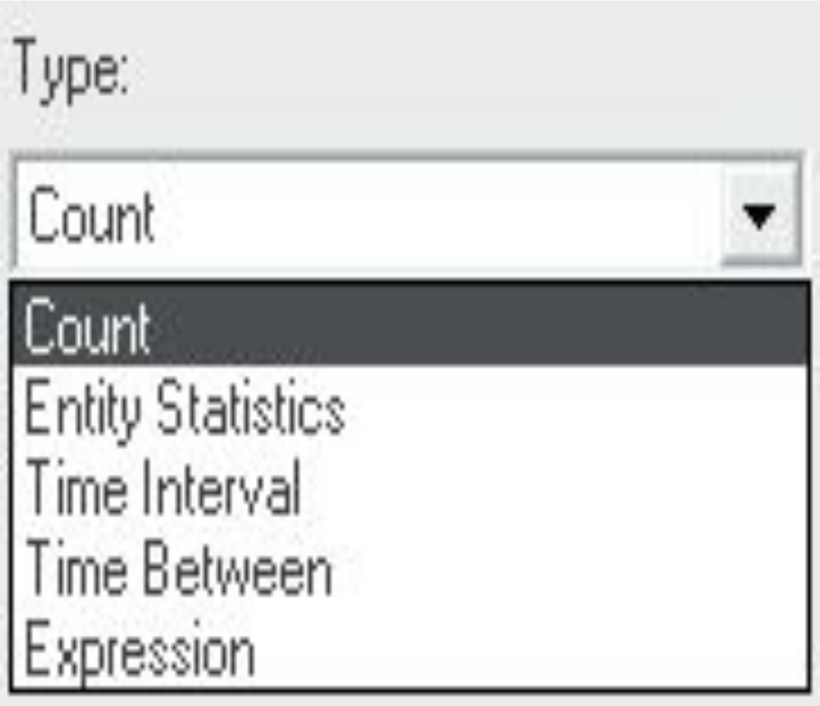
Value: 1

□ Record into Set

Counter Name: Record 1

- Name
- Type
- Value
- Counter Name
- Record into Set

Record module parameter: Type



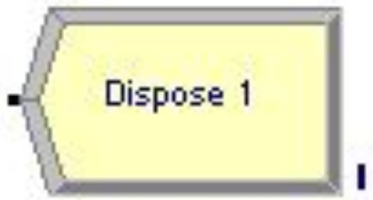
Type:

Count ▼

Count
Entity Statistics
Time Interval
Time Between
Expression

- Count
- Entity Statistics
- Time Interval
- Time Between
- Expression

1.1.8 Dispose

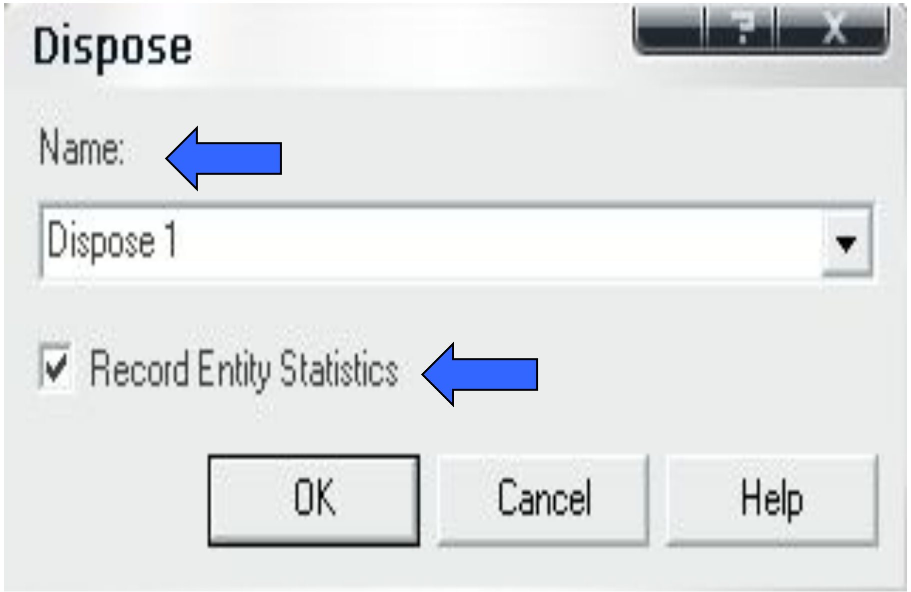


Dispose module is end point for entity leaving from simulating model

Application of *Dispose* module

- □ Documents processed
- □ Clients come out

Dispose module parameter



Dispose

Name:

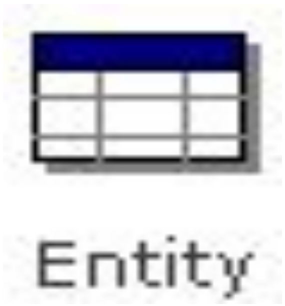
Dispose 1

☑ Record Entity Statistics

- Name
- Record Entity Statistics

1.2 Data Modules

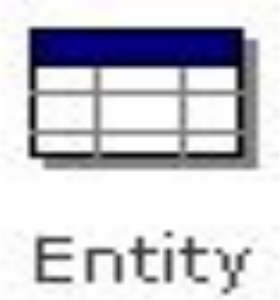
1.2.1 Entity



Entity module sets entity type and entity initial picture in the model

For every *Create* module have to define entity type which will be generated

Application of *Entity* module



□ Documents: faxes, letters, reports and etc.

□ People: workers, managers, men and etc.

Entity module parameters

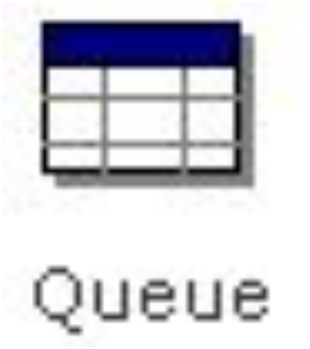
- Entity type
- Initial picture

1.2.2 Queue

Queue module uses for setting of queue types:

- First in First out (FIFO)
- Last in first out (LIFO)
- Lowest Attribute Value
- Highest Attribute Value

Application of *Queue* module



- □ Queue in the supermarket where people are waiting cashier service

- □ Queue of parts are waiting machining

1.2.3 Resource

Resource module allows to set recourse which is associated with certain process

Application of *Resource* module

- □ People: workers, managers, men, salespeoples and etc.
- □ Equipment: telephone loop, machines, computers and etc.

Resource module parameters

- Resource type (Fixed Capacity, Based on Schedule)
- Capacity

1.2.4 Schedule

Schedule module can be uses for setting of the time interval:

1. Generation of entities in the model (*Create* module)
2. Processing of entities in the model (Process module)

Application of *Schedule* module

□ Time-table of staff

□ Number of buyers are arrived at supermarket at certain time intervals

Schedule module parameter

- □ Type
  - ✔ Capacity (*Process* module),
  - ✔ Arrival (*Create* module)
  - ✔ Other

- □ Time units

1.2.5 Set

Set module is defined recourse set, which will be associated with *Process*

Set module parameter

- Members
  - Cyclical
  - Preferred Order
- Resource Name

1.2.6 Variable

Variable

Variable module defines variable names and variable initial values

Application of *Variable* module

- □ To count a number of documents which were processed during certain time interval
- □ To identify serial number of element

Variable module parameter

- Initial Value
- Rows
- Columns
- Clear Option
  - Statistics
  - System
  - None
- Statistics

Thank you for attention!